International Cartographic Association
Association Cartographique Internationale

Automatic vectorization of point symbols on archive maps using deep convolutional neural network

Gergely Vassányi [a,*] , Mátyás Gede [a]

[a] *Eötvös Loránd University Budapest, Department of Cartography and Geoinformatics, Gergely Vassányi – vassanyigergely@gmail.com, Mátyás Gede – saman@map.elte.hu*

* Corresponding author

Abstract: Archive topographical maps are a key source of geographical information from past ages, which can be valuable for several science fields. Since manual digitization is usually slow and takes much human resource, automatic methods are preferred, such as deep learning algorithms. Although automatic vectorization is a common problem, there have been few approaches regarding point symbols. In this paper, a point symbol vectorization method is proposed, which was tested on Third Military Survey map sheets using a Mask Regional Convolutional Neural Network (MRCNN). The MRCNN implementation uses the ResNet101 network improved with the Feature Pyramid Network architecture and is developed in a Google Colab environment. The pretrained network was trained on four point symbol categories simultaneously. Results show 90% accuracy, while 94% of symbols detected for some categories on the complete test sheet.

Keywords: MRCNN, vectorization, map symbols, Python

1. Introduction

Historical maps contain vast amounts of geographical information, which can help us understand environmental and physical changes that happened over past ages. These maps, however, are usually in printed form and do not have a vector geo-data representation, which is critical in modern geospatial analysis since these methods are mostly digital. Creating vector data by manual digitization – especially with hundreds of sheets – is mostly a slow and laborious process; therefore, it is profitable to use automatic solutions where possible.

There have been several approaches to this problem in recent years. Iosifescu et al. proposed a vectorization methodology for extracting areal and linear features using binary image segmentation and vectorizing tools provided by the open source GDAL and OGR libraries (Iosifescu, Tsorlini, and Hurni 2016). A similar study by Gede et al. focused on vectorizing linear hydrographic features using QGIS (Gede et al. 2020). These studies also provide scanning and rectifying techniques as well as several raster pre-processing and vector cleaning methods.

In case of point symbols, vectorization can be viewed as an object detection task where the goal is to identify the type of the object and obtain the coordinates of its geographical reference, which can be stored in a spatial database. Supervised machine learning algorithms, such as Deep Convolutional Neural Networks (CNN) have proved to be more effective for such object detection tasks than conventional segmentation methods. CNN-s have been used in various map vectorization applications, such as automatic label extraction (Laumer et al. 2020), wetland extraction (Jiao, Heitzler, and Hurni 2020) and improved image segmentation by predictions of areal symbol locations (Groom et al. 2020). Saeedimoghaddam and Stepinski used CNN to detect road intersection points and achieved an average of 90% accuracy, with 82% of intersection points extracted (Saeedimoghaddam and Stepinski 2020). Quan et al. focused on point symbol recognition combined with image segmentation on a pretrained CNN of AlexNet architecture and achieved 98.97% accuracy on single test images (Quan et al. 2018).

In this study, an improved CNN was used to detect four distinct point symbols simultaneously on scanned historical topographic maps. The process is based on transfer learning using a pretrained network (Tan et al. 2018), which shortens the training time and gives promising results with a relatively small dataset. Detection can be applied to large images (i.e. complete scanned map sheets), and resulting points are transformed into projection coordinates. The achieved detection accuracy is 90%, with 94% of symbols detected for some categories on the test sheet. Final results are visualized by GIS Software.

2. Data and methods

2.1 Target map

The target maps of this study were scanned versions of four 1 : 200 000 scale map sheets from the Third Military Survey of Austria-Hungary (made at the end of the XIX. century). These sheets can be found physically in the Map Collection of ELTE Eötvös Loránd University. The sheets cover a $1° \times 1°$ area of the Earth and depict territories of present Romania and Bulgaria, near the Danube river. The terrain is diverse, including mountainous regions, hills, and lowlands; therefore, it is

30th International Cartographic Conference (ICC 2021), 14–18 December 2021, Florence, Italy. This contribution underwent single-blind peer review based on submitted abstracts. https://doi.org/10.5194/ica-proc-4-109-2021

suitable to experiment on symbol detection over different backgrounds.

There are several point symbol categories depicted on the sheets. In this paper, the focus was on frequently shown symbols because training a neural network requires a sufficient amount of training and validating images that contain the target object. For some categories, the appearance of the symbols is different between the sheets, mainly because of the various styles of drawing used. Considering the above, four categories with uniform symbols were chosen as subject to detection, which are: catholic church, orthodox church, elevation point, and watermill (Figure 1).

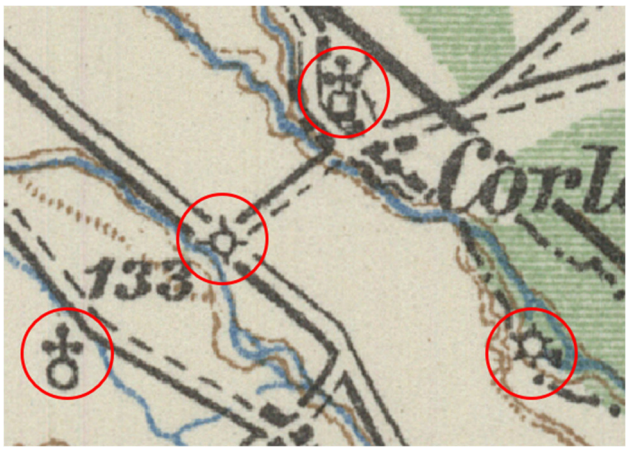

Figure 1. The symbols subject to detection. From left to right: catholic church, elevation point, orthodox church, watermill.

2.2 Neural network

The deep neural network used for symbol detection in this study is a Mask Regional Convolutional Neural Network, or MRCNN (He et al. 2020). The MRCNN is designed for instance segmentation purposes, combining object detection and semantic segmentation. This means generating bounding boxes and segmentation masks for each instance of an object, such as people, balloons, or map symbols.

In case of vectorizing point symbols, the segmentation mask is not that relevant because we are only interested in the coordinates of the symbol's centre, which can be calculated merely from the bounding box. Nevertheless, when detecting areal features, such as lakes or settlements, the mask shows the objects' exact extension, making it an essential part of the process.

The open source implementation of MRCNN used in this study can be found in a repository on GitHub (Abdulla 2017). The model uses the ResNet101 network as backbone (He et al. 2016), which is a standard convolutional neural network that serves as a feature extractor. This is improved with a Feature Pyramid Network architecture which performs better at representing objects at multiple scales than Single Feature Map approaches (Lin et al. 2017).

The repository contains pretrained weights on the MS COCO dataset, a large, richly annotated dataset containing common objects (Lin et al. 2014). This means that these weights can be used as a starting point to train the model with one's own dataset, simplifying the training process. The source code of this MRCNN build is written in Python3 and utilizes the Keras and TensorFlow libraries. Programs for training, debugging, and visualization are also available in Jupyter Notebook form. Source code of the program discussed in this paper is available at:

https://github.com/vassanyig/custom_mrcnn/tree/master/4symbols_code.

2.3 Training data

Training the model requires manually annotated images of the target objects. These images were produced by cutting small territories from the original map and then annotated with an open source web tool called Make Sense (Skalski 2019). To serve as input for the neural network, the images were divided into training and validation groups in a 70 : 30 ratio due to the small size of the dataset. For all four symbols, a hundred of images were used, although some of them contain more than one annotated object due to the density of the symbols on the original map. The annotated symbols are from various parts of the map sheets, covering many different backgrounds. The size of symbols on the images can also change, which is handled due to the multi-scale approach of the Feature Pyramid Network.

2.4 Colab

The detecting algorithm is ran in a virtual environment called Colaboratory (or Colab), a free service provided by Google (Nelson and Hoover 2020). Colab offers a Linux-based virtual machine with 8-12 GB RAM and 50-70 GB hard drive space, and a GPU that can be utilized to accelerate the computing process. The code is in Jupyter Notebook form, which allows interactivity when running the program. Being a Google service, Colab supports Google Drive connection, which is a convenient way of importing and exporting files from the process, such as trained models, target images, and results. The runtime is limited to 12 hours; therefore, setting up the workspace and establishing the Google Drive connection has to be done at every session.

2.5 Training the model

To train the model, the pretrained weights file and the annotated images have to be provided to the code, which was done through Google Drive. The code was modified to include four classes with their label names used in the annotation process. Configuration parameters can be manually set, such as minimum detection confidence and the number of epochs. In the created model, all layers were trained with four epochs using 1500 training steps per epoch. Additionally, a minimum of 80 percent confidence was specified. The trained model is saved in a Google Drive folder, which makes it an easy access for the detection program.

30th International Cartographic Conference (ICC 2021), 14–18 December 2021, Florence, Italy. This contribution underwent single-blind peer review based on submitted abstracts. https://doi.org/10.5194/ica-proc-4-109-2021 |

2.6 Detection

Before running the detection script, paths to the trained model and the target images must be specified. The configuration parameters of the detection should be the same as the ones used for training. After running the detection, results can be visualized by a module provided in the source code, which draws the detected features (e. g. bounding boxes, segmentation masks, labels etc.) on the target image. Although this can be useful for illustration, the output is still just a raster image.

To obtain vectoral information about the detected symbols, bounding boxes are used to calculate the coordinates of their geographical reference points. In case of the elevation point and the watermill symbols, this reference point is at the centre of the symbols, meaning that the centre of the bounding box can be used. However, the reference point of the catholic and orthodox churches is the centre of the bottom circle or rectangle, respectively. For these symbols, the reference point is put lower than the middle in the Y-axis by one fourth of the symbol's total length.

The exported coordinates of the symbols are pixel coordinates of the target image, which are written into a CSV file. For each coordinate pair, the label of the object (i.e., it's class) and the confidence score are included.

2.7 Detection on large images

In the original version of the code, all target images are resized to 1024×1024 pixels, which shrinks larger images, and consequently, the efficiency of the algorithm decreases. When image resizing is disabled, test results showed that running the detection on large images significantly increases the runtime to a point where the program results in an error. To work around this problem, an image splitting Python script has been written that makes tiles of the original image in the desired size with arbitrary overlap. These images are fed one by one to the detecting algorithm, and the resulting CSV files are then merged by another program into a final CSV that contains all detected symbols with pixel coordinates of the original split image. Objects on overlapping parts can be detected multiple times; therefore, the program filters points closer to each other than a threshold, supposing that they represent the same object.

2.8 Georeferencing

Additionally, a function was added to the CSV file merging program that transforms pixel coordinates into projection coordinates. This is only possible if the original image is georeferenced and a world file (.TWF) is provided, which can be automatically generated when georeferencing the image. The end results can be viewed by GIS Software and could be exported in another format, such as Esri Shapefile (.SHP).

3. Results

3.1 Single images

The detecting program was run on single images cut from the original map in different scales of zoom. These files were not georeferenced as they served testing purposes to set the model parameters to achieve the best results. On these images, the model detected most of the symbols correctly without any cases of mixing them up or falsely detecting a symbol for a different object. Results were displayed using MRCNN's visualize module (Figure 2). Some symbols were not detected, which was present in three of the four categories, but not with the catholic church symbol. Test results also showed that there is a dependency on the size of the symbols in terms of efficiency of the detection algorithm. In general, the model performed better at finding large symbols than small ones.

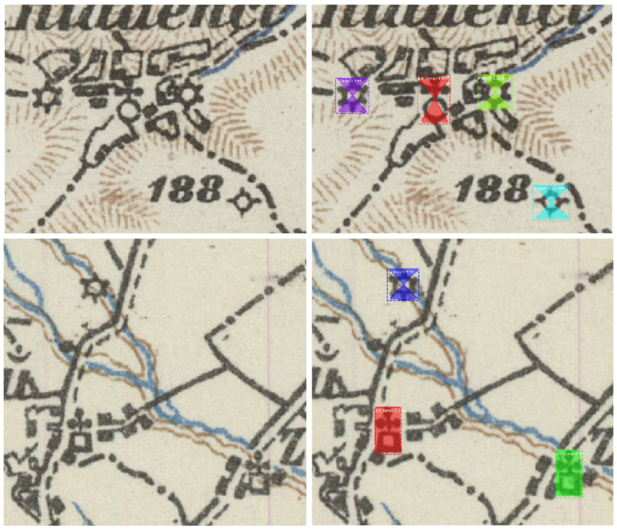

Figure 2. Examples of the results on single images. Left: target image, right: resulting image with masks and bounding boxes for each symbol. Different colours represent different instances.

3.2 Complete sheet

The algorithm was run on a complete map sheet, namely the "43°44° Svistov" sheet scanned with 600 dpi resolution and 24-bit colour depth. The scanned map was saved in JPG format, having the extents of 10 215 pixels in width and 14 352 pixels in length. The image was georeferenced using Global Mapper in WGS84 projection, and a .TFW file was generated. The image was then split with the python program, where rectangular tiles were chosen with 1024-pixel tile size. An overlap of 8% was also set to avoid failed detection due to symbols split in half. The detection algorithm was then run on the tiles, and the outputs were combined into a single file with projection coordinates of the points using the CSV merging program. Results were visualized as point symbols added as a vector layer over the georeferenced raster file using QGIS (Figure 3).

The results on the Svistov sheet show significant differences between the symbol categories (Table 1). In case of the catholic church, 326 instances were detected, of which 33 proved to be false-positive. There were also 16 symbols the detection algorithm missed. The remaining 293 symbols were correctly detected, which

shows an 89.88% precision and covers 94.82% of the total number of symbols on the sheet.

The algorithm proved less effective on the elevation points with only four false-positive cases out of 161 detections, but more than half of all the symbols missed. In case of the watermill and the orthodox church, the program was not able to detect any of the symbols on the map. This is not so surprising with the orthodox church because there are only two occurrences of this symbol on the sheet, but there are many watermills the model should have been able to detect.

Symbol	Catholic church	Elev. point	Watermill	Orth. church
Total count	309	336	99	2
Total detections	326	161	No detections	
False positives	33	4		
Undetected	16	179		
Correct detections	293	157		
Precision	89.88%	97.52%		
Symbols detected	94.82%	46.73%		

Table 1. Results of the detection process for each symbol.

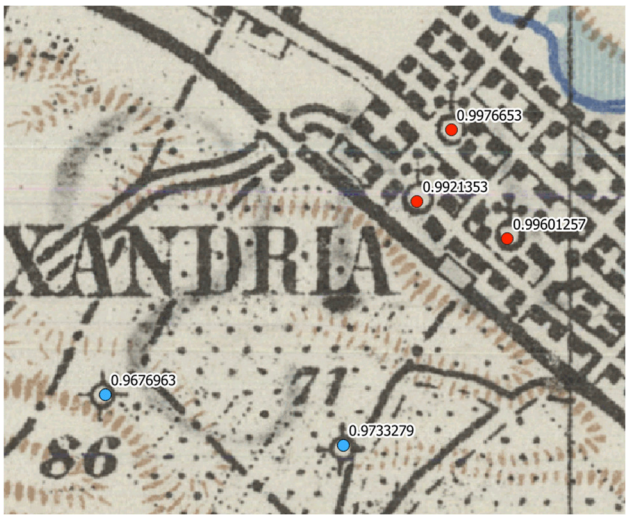

Figure 3. An example of the detected features as vector points drawn over the original map. Red: catholic church; Blue: elevation point. Numbers represent confidence scores for each point.

4. Conclusions

4.1 Discussion of results

The parameters of the model were optimized on single images, and, in general, the model detected the symbols sufficiently on them. However, the same model was less effective on a complete sheet, maintaining the detection accuracy for only one of the four symbols. Differences in the Svistov sheet results indicate that the model was well trained on the catholic church symbol and less adequately on the other three categories, which might come from the training images used.

The majority of these images contain larger symbols in pixel size than the ones on the sheet. This could have influenced the neural network to become more sensitive for this symbol extent rather than the size of the symbols on the input. Since the training images were made by hand in various zoom and extent, it is likely that the way they were created affects the model to work better for the catholic church and worse for the other three categories. A possible way to overcome this problem is to create training images for all categories using the same zoom as the target map, and then retrain the neural network.

In some cases of false symbol detections, the detected object was a different, similarly looking symbol, which the model had not been trained with (Figure 4). The confidence score of these features is somewhat lower than the score of the correct guesses, but still over the specified threshold. These detections happen when there is no information provided to the model that the object represents a different category; therefore, the model classifies it as the most likely one of the trained categories. Cases like this could be reduced by training the model with more symbols, which is only limited by the availability of training images.

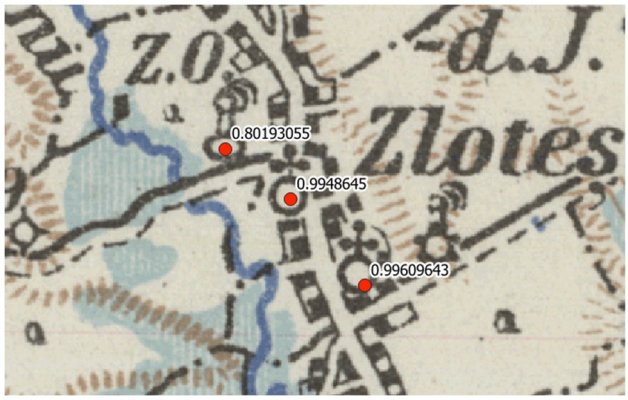

Figure 4. False detection on an untrained symbol (chimney, on the left). The confidence score is just above the 0.8 threshold, whereas it is almost 1 for the correctly identified churches.

Comparing the results to other studies, such as the study of Saeedimoghaddam and Stepinski shows that similar precision and detection rates could be achieved using this methodology, and with the correction of the training images and algorithm, efficiency can be further increased.

4.2 Conclusion

Results showed that this methodology could be used effectively to create vector point features from scanned raster map symbols, where different categories are detected simultaneously, and the results can be visualized easily by GIS software. The process also works with large, georeferenced images, but the results are not perfect, and further improvement is needed.

Future plans include refining the experimentally chosen parameters of the training configuration in order to

achieve higher efficiency. Retraining the model with images containing the symbols in the same size as on the target map could help to achieve the same detection accuracy for all categories. Another field of interest is adding more symbols to the training process, which would significantly increase the amount of information extracted from the map.

4.3 Acknowledgement

EFOP-3.6.3-VEKOP-16-2017-00001: Talent Management in Autonomous Vehicle Control Technologies. The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

5. References

Abdulla, Waleed. 2017. "Mask R-CNN for Object Detection and Instance Segmentation on Keras and TensorFlow." *GitHub Repository*. Github. https://github.com/matterport/Mask_RCNN.

Gede, Mátyás, Valentin Árvai, Gergely Vassányi, Zsófia Supka, Enikő Szabó, Anna Bordács, Csaba Gergely Varga, and Krisztina Irás. 2020. "Automatic Vectorisation of Old Maps Using QGIS –Tools, Possibilities and Challenges." In *International Workshop on Automatic Vectorisation of Historical Maps-13 March 2020 -ELTE, Budapest*. doi:10.21862/avhm2020.04.

Groom, Geoff, Gregor Levin, Stig Svenningsen, and Mads Linnet Perner. 2020. "Historical Maps – Machine Learning Helps Us over the Map Vectorisation Crux." In . doi:10.21862/avhm2020.11.

He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2020. "Mask R-CNN." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2). IEEE Computer Society: 386–397. doi:10.1109/TPAMI.2018.2844175.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep Residual Learning for Image Recognition." In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2016-December. doi:10.1109/CVPR.2016.90.

Iosifescu, Ionut, Angeliki Tsorlini, and Lorenz Hurni. 2016. "Towards a Comprehensive Methodology for Automatic Vectorization of Raster Historical Maps." *E-Perimetron* 11 (2).

Jiao, Chenjing, Magnus Heitzler, and Lorenz Hurni. 2020. "Extracting Wetlands from Swiss Historical Maps with ConvolutionalNeural Networks." In *International Workshop on Automatic Vectorisation of Historical Maps-13 March 2020 -ELTE, Budapest*. doi:10.21862/avhm2020.03.

Laumer, Daniel, Hasret Gümgümcü, Magnus Heitzler, and Lorenz Hurni. 2020. "A Semi-Automatic Label Digitization Workflow for the Siegfried Map." In *International Workshop on Automatic Vectorisation of Historical Maps-13 March 2020 -ELTE, Budapest*. doi:10.21862/avhm2020.07.

Lin, Tsung Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. "Feature Pyramid Networks for Object Detection." In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Vol. 2017-January. doi:10.1109/CVPR.2017.106.

Lin, Tsung Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. "Microsoft COCO: Common Objects in Context." In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8693 LNCS. doi:10.1007/978-3-319-10602-1_48.

Nelson, Mark J., and Amy K. Hoover. 2020. "Notes on Using Google Colaboratory in AI Education." In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*. doi:10.1145/3341525.3393997.

Quan, Yining, Yuanyuan Shi, Qiguang Miao, and Yutao Qi. 2018. "A Combinatorial Solution to Point Symbol Recognition." *Sensors (Switzerland)* 18 (10). doi:10.3390/s18103403.

Saeedimoghaddam, Mahmoud, and T. F. Stepinski. 2020. "Automatic Extraction of Road Intersection Points from USGS Historical Map Series Using Deep Convolutional Neural Networks." *International Journal of Geographical Information Science* 34 (5). doi:10.1080/13658816.2019.1696968.

Skalski, Piotr. 2019. "Make Sense."

Tan, Chuanqi, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. "A Survey on Deep Transfer Learning." In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 11141 LNCS. doi:10.1007/978-3-030-01424-7_27.